Actividad N° 2 - Ejercicios de Funciones

Alumno: Cristhian Arlindo Mamani Nina

Universidad Nacional del Altiplano Puno
Facultad de Ingeniería, Estadística e Informática

Objetivo de la Actividad

El objetivo de esta actividad es resolver una serie de problemas que modelan relaciones lineales en diversos contextos, con el propósito de aplicar los conceptos de optimización y analizar el comportamiento de las variables mediante fórmulas y gráficas generadas a través de código Python.

Ejercicio 1: Cálculo del Precio de una Vivienda

Concepto: El precio de una vivienda (P) depende linealmente del área construida (A), y se puede expresar mediante la siguiente fórmula:

$$P = mA + b \tag{1}$$

Donde:

- m es el costo por metro cuadrado de construcción.
- A es el área construida de la vivienda.
- b son los costos fijos asociados.

Gráfica y Código en Python: La relación entre el área construida y el precio de la vivienda se puede representar mediante una gráfica generada por el siguiente código:

```
import numpy as np
import matplotlib.pyplot as plt

# Funci n para calcular el precio
def calcular_precio_vivienda(m, A, b):
    return m * A + b

# Funci n para graficar el precio en funci n del  rea  construida
def graficar_precio_vivienda(m, b, A_min, A_max):
    A = np.linspace(A_min, A_max, 100)
```

```
    P = calcular_precio_vivienda(m, A, b)
    plt.plot(A, P, label=f"m={m}, b={b}")
    plt.xlabel(" rea  construida (A)")
    plt.ylabel("Precio (P)")
    plt.title("Precio de Vivienda vs  rea  Construida")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 1: Código para calcular el precio de la vivienda

Ejercicio 2: Cálculo de la Ganancia Mensual

Concepto: La ganancia mensual (G) de un modelo depende linealmente del número de predicciones realizadas (N), y se expresa como:

$$G = cN + b \tag{2}$$

Donde:

- c es la ganancia por predicción.
- N es el número de predicciones realizadas.
- b son los ingresos fijos mensuales.

Gráfica y Código en Python: La relación entre el número de predicciones realizadas y la ganancia mensual se puede visualizar con el siguiente código:

```
def calcular_ganancia_mensual(c, N, b):
    return c * N + b

def graficar_ganancia_mensual(c, b, N_min, N_max):
    N = np.linspace(N_min, N_max, 100)
    G = calcular_ganancia_mensual(c, N, b)
    plt.plot(N, G, label=f"c={c}, b={b}")
    plt.xlabel("N mero de predicciones (N)")
    plt.ylabel("Ganancia (G)")
    plt.title("Ganancia Mensual vs N mero de Predicciones")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 2: Código para calcular la ganancia mensual

Ejercicio 3: Cálculo del Tiempo de Procesamiento

Concepto: El tiempo total de procesamiento (T) depende linealmente del tamaño de los datos (D), y se expresa como:

$$T = kD + c \tag{3}$$

Donde:

- k es el tiempo de procesamiento por unidad de dato.
- D es el tamaño de los datos a procesar.
- c es el tiempo de procesamiento fijo.

Gráfica y Código en Python: El comportamiento del tiempo de procesamiento frente al tamaño de los datos se puede modelar con el siguiente código:

```
def calcular_tiempo_procesamiento(k, D, c):
    return k * D + c

def graficar_tiempo_procesamiento(k, c, D_min, D_max):
    D = np.linspace(D_min, D_max, 100)
    T = calcular_tiempo_procesamiento(k, D, c)
    plt.plot(D, T, label=f"k={k}, c={c}")
    plt.xlabel("Tama o de datos (D)")
    plt.ylabel("Tiempo (T)")
    plt.title("Tiempo de Procesamiento vs Tama o de Datos")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 3: Código para calcular el tiempo de procesamiento

Ejercicio 4: Cálculo del Costo Total de Almacenamiento

Concepto: El costo total (C) depende linealmente de la cantidad de datos almacenados (D) y se expresa mediante la fórmula:

$$C = pD + f \tag{4}$$

Donde:

- p es el costo por unidad de datos almacenados.
- D es la cantidad de datos almacenados.
- f son las tarifas fijas asociadas al almacenamiento.

Gráfica y Código en Python: Para calcular el costo total de almacenamiento en función de la cantidad de datos, se utiliza el siguiente código:

```
def calcular_costo_total(p, D, f):
    return p * D + f

def graficar_costo_total(p, f, D_min, D_max):
    D = np.linspace(D_min, D_max, 100)
    C = calcular_costo_total(p, D, f)
    plt.plot(D, C, label=f"p={p}, f={f}")
    plt.xlabel("Cantidad de datos (D)")
    plt.ylabel("Costo Total (C)")
    plt.title("Costo Total vs Cantidad de Datos")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 4: Código para calcular el costo total de almacenamiento

Ejercicio 5: Cálculo de Medición Calibrada de un Sensor

Concepto: La medición calibrada (M) de un sensor depende linealmente de la medición en crudo (R) y se expresa como:

$$M = aR + b \tag{5}$$

Donde:

- a es el factor de ajuste.
- R es la medición en crudo.
- b es el desplazamiento constante.

Gráfica y Código en Python:

```
def calcular_medicion_calibrada(a, R, b):
    return a * R + b

def graficar_medicion_calibrada(a, b, R_min, R_max):
    R = np.linspace(R_min, R_max, 100)
    M = calcular_medicion_calibrada(a, R, b)
    plt.plot(R, M, label=f"a={a}, b={b}")
    plt.xlabel("Medici n en crudo (R)")
    plt.ylabel("Medici n Calibrada (M)")
    plt.title("Medici n Calibrada vs Medici n en Crudo")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 5: Código para calcular la medición calibrada

Ejercicio 6: Cálculo del Tiempo de Respuesta Promedio de un Servidor

Concepto: El tiempo de respuesta promedio (T) de un servidor depende linealmente del número de solicitudes simultáneas (S) y se expresa como:

$$T = mS + b \tag{6}$$

Donde:

- m es el incremento del tiempo por solicitud.
- S es el número de solicitudes simultáneas.
- b es el tiempo base del servidor.

Gráfica y Código en Python: La relación entre el número de solicitudes simultáneas y el tiempo de respuesta promedio se puede modelar con el siguiente código:

```
def calcular_tiempo_respuesta(m, S, b):
    return m * S + b

def graficar_tiempo_respuesta(m, b, S_min, S_max):
    S = np.linspace(S_min, S_max, 100)
    T = calcular_tiempo_respuesta(m, S, b)
    plt.plot(S, T, label=f"m={m}, b={b}")
    plt.xlabel("Solicitudes simult neas (S)")
    plt.ylabel("Tiempo de Respuesta (T)")
    plt.title("Tiempo de Respuesta vs Solicitudes Simult neas")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 6: Código para calcular el tiempo de respuesta promedio

Ejercicio 7: Cálculo de Ingresos de una Plataforma

Concepto: Los ingresos (I) de una plataforma dependen del número de suscriptores (S) y se expresan como:

$$I = pS + b \tag{7}$$

Donde:

- p es el ingreso promedio por suscriptor.
- S es el número de suscriptores.
- b son los ingresos adicionales fijos.

Gráfica y Código en Python:

```
def calcular_ingresos(p, S, b):
    return p * S + b

def graficar_ingresos(p, b, S_min, S_max):
    S = np.linspace(S_min, S_max, 100)
    I = calcular_ingresos(p, S, b)
    plt.plot(S, I, label=f"p={p}, b={b}")
    plt.xlabel("N mero de suscriptores (S)")
    plt.ylabel("Ingresos (I)")
    plt.title("Ingresos vs N mero de Suscriptores")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 7: Código para calcular los ingresos

Ejercicio 8: Cálculo de Energía Consumida

Concepto: La energía consumida (E) depende linealmente del número de operaciones realizadas (O) y se expresa como:

$$E = kO + b \tag{8}$$

Donde:

- k es la energía consumida por operación.
- O es el número de operaciones realizadas.
- b es la energía base consumida por el sistema.

Gráfica y Código en Python:

```
def calcular_energia_consumida(k, O, b):
    return k * O + b

def graficar_energia_consumida(k, b, O_min, O_max):
    O = np.linspace(O_min, O_max, 100)
    E = calcular_energia_consumida(k, O, b)
    plt.plot(O, E, label=f"k={k}, b={b}")
    plt.xlabel("Operaciones realizadas (O)")
    plt.ylabel("Energ a Consumida (E)")
```

```
    plt.title("Energ a Consumida vs Operaciones Realizadas")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 8: Código para calcular la energía consumida

Ejercicio 9: Cálculo de Likes en una Publicación

Concepto: El número de likes (L) en una publicación depende linealmente del número de seguidores (F) y se expresa como:

$$L = mF + b \tag{9}$$

Donde:

- m es la proporción promedio de interacción.
- F es el número de seguidores de la publicación.
- b es el nivel base de likes.

Gráfica y Código en Python:

```
def calcular_base_seguidores(m, F, b):
    return m * F + b

def graficar_base_seguidores(m, b, F_min, F_max):
    F = np.linspace(F_min, F_max, 100)
    L = calcular_base_seguidores(m, F, b)
    plt.plot(F, L, label=f"m={m}, b={b}")
    plt.xlabel("N mero de Seguidores (F)")
    plt.ylabel("Base de Likes (L)")
    plt.title("Base de Likes vs N mero de Seguidores")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 9: Código para calcular la base de seguidores

Ejercicio 10: Cálculo del Costo Total para Entrenar un Modelo

Concepto: El costo total (C) para entrenar un modelo depende del número de iteraciones (t) y se expresa como:

$$C = pt + c \tag{10}$$

Donde:

- p es el costo por iteración.
- t es el número de iteraciones.
- c son los costos iniciales de entrenamiento.

Gráfica y Código en Python:

```
def calcular_costo_ml(p, t, c):
    return p * t + c

def graficar_costo_ml(p, c, t_min, t_max):
    t = np.linspace(t_min, t_max, 100)
    C = calcular_costo_ml(p, t, c)
    plt.plot(t, C, label=f"p={p}, c={c}")
    plt.xlabel("N mero de Iteraciones (t)")
    plt.ylabel("Costo Total (C)")
    plt.title("Costo Total vs N mero de Iteraciones")
    plt.legend()
    plt.grid()
    plt.show()
```

Listing 10: Código para calcular el costo total de entrenamiento

Conclusión General

Cada uno de los ejercicios presentados demuestra cómo las relaciones lineales pueden ser aplicadas en diferentes contextos para modelar fenómenos reales. Además, se ha utilizado la programación en Python para realizar los cálculos y generar las gráficas correspondientes, lo que facilita la visualización de los resultados obtenidos. Este enfoque permite a los estudiantes comprender mejor el comportamiento de las variables involucradas y cómo se pueden optimizar los procesos analizados.